International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **12**, December **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 8311–8321

SELF-EVOLVING ANT COLONY OPTIMIZATION AND ITS APPLICATION TO TRAVELING SALESMAN PROBLEM

Xiao-Fan Zhou and Rong-Long Wang

Graduate School of Engineering
University of Fukui
Bunkyo 3-9-1, Fukui-shi 910-8507, Japan
wang@u-fukui.ac.jp

Received September 2011; revised January 2012

Abstract. *Ant colony optimization (ACO) algorithms are a recently developed, population-based approach which has been successfully applied to combinatorial optimization problems. However, in the ACO algorithms, it is difficult to adjust the balance between intensification and diversification and thus the performance is not always very well. In this paper we proposed an improved ACO algorithm in which a self-evolving strategy is adopted so as to acquire a trade-off between intensification and diversification. With this strategy we make each ant evolve by means of multiple pheromone deposition, the total concentration of which includes two parts, that is, one part is deposited by several best ants of the last iteration and the ant that has searched the global-best solution by the current iteration, and the other part is deposited temporarily by each ant itself of the last iteration as well. The proposed algorithm is tested by simulating Traveling Salesman Problem (TSP) and experimental results show that Self-evolving Ant Colony Optimization has superior performance when compared with other existing ACO algorithms.*
Keywords: Ant colony optimization, Combinatorial optimization problems, Traveling salesman problem, Self-evolving strategy

1. **Introduction.** ACO is a recently developed meta-heuristic method [1,2] that has successfully applied to a number of combinatorial optimization problems arising in many different fields such as economy, commerce, engineering, industry or medicine [3-7]. The inspiring source of ACO algorithms is the foraging behaviors of real ants which deposit a chemical pheromone trail on the ground to mark the favorable path that should be followed by other ants of the colony when searching for foods. In ACO, a number of artificial ants build solutions to the considered combinatorial optimization problem and exchange information on the quality of these solutions by means of an indirect communication which enables them to find short paths between their nest and food sources using pheromone trails. Compared with real ants, ACO algorithms exploit a similar mechanism for solving, for example, discrete optimization problems.

The searching behavior of ACO can be characterized by two main features [8-10]: intensification and diversification. Intensification is the ability to search thoroughly in the local neighborhood where good solutions have previously been found, while diversification is the ability to broadly search through the solution space. Higher intensification leads to the rapid convergence to a suboptimal solution while higher diversification results in a better solution at higher computational cost due to the slow convergence of the approach. Since these attributes are in conflict with each other, a trade-off between them is important for a logical balance between the efficiency and optimality. Since the first ACO algorithm called Ant System (AS) [11-13] was proposed by Dorigo in 1992, a number of other ACO algorithms were introduced. The first improvement over AS was

obtained by the Elitist AS (EAS) [14], which has a modified pheromone update rule, that is, each time the pheromone trails are updated, those belonging to the edges of the global best tour get an additional amount of pheromone so as to emphasize information about the best solution in the algorithms' search procedure. The biggest problem that can be caused by such exploitative method is insufficient exploration and premature convergence to sub-optimal solutions. As a result, different remedies using exploitative methods, such as EAS, are proposed to solve the premature convergence problem. In Rank-based version of AS (Rank-based AS) [15], the concept of ranking was applied and extended as follows. A number of best ants of the current iteration are allowed to deposit pheromone according to their ranks, while the best-so-far ants are allowed to deposit pheromone of the highest quantity. Another improvement over Ant System is Max-Min Ant System (MMAS) [16-18], of which the characterizing elements are that it also adopts a concept of elitism in which only the best ant at each iteration updates trails and that the possible trail values are restricted to the interval $[\tau_{\min}, \tau_{\max}]$, where the two parameters are set in a problem dependent way.

Ant Colony System (ACS) [19,20] introduced by Dorigo uses a more aggressive action choice rule than AS, called pseudo random proportional rule which favors transitions towards nodes connected by short edges and with a large amount of pheromone. In ACS the pheromone updating rule is only applied to the edges belonging to the global-best tour, and a local pheromone updating rule, by which each time an ant uses an edge (i, j) to move from city i to city j, it removes some pheromone from the edge, is also adopted.

Although modifications on preventing ant from converging to local optimum through exploiting new pheromone update rules have been carried out and acquired relatively good performance, the balance between intensification and diversification is still the most important theme in the study of ACO algorithms.

In this paper, focusing on the balance between intensification and diversification, we proposed a new anti-stagnation method, called self-evolving ACO algorithm, to which the rank-based strategy of pheromone update in ACO algorithms is applied so as to concentrate the search around the good solutions as a means of encouraging the intensification of the method. On the other hand, together with the rank-based pheromone, the pheromone related with the result that each ant found in the last iteration is also introduced as a new way of employing pheromone to enhance the diversification. By this method, the result of each ant evolves iteration by iteration. AS pheromone of each ant itself played a vital role in solution construction, we call the proposed algorithm self-evolving Ant Colony Optimization in this paper.

To evaluate the performance of the proposed improved ACO algorithm, we simulate some TSPLIB benchmark problems [21,22]. The simulation results show that the proposed algorithm performs better than the existing ACO algorithms on some TSP benchmark problems.

Since the first ACO was proposed, a lot of improved versions of it were introduced and successfully applied to various combinatorial optimization problems arising in many different fields. Many studies on the ACO algorithms were conducted and extensive experiments were implemented on different benchmark problems such as TSP, and the theoretical results can be widely applied to many real-world applications. For instance, in the order-picking problem in warehouses, some vehicle has to collect all items of an order that contains a certain subset of the items stored in the warehouse, and ship them to the customer. The problem of finding a shortest route for the vehicle with minimum pickup time can be solved by the ACO algorithms. Besides, the scheduling problem in the manufacturing plant can be also solved by a practical application using ACO algorithm. The objective of the problem is to find a way to successfully manage resources in order

to produce products in the most efficient way possible and design a production schedule that promotes on-time delivery, and minimizes objectives such as the flow time of a product. In addition, other practical uses of the ACO algorithms include the drilling problem of printed circuit boards, the facility layout problems, the routing problems in telecommunication networks and other optimal problems.

2. **Ant Colony Optimization.** The first ACO algorithm, called Ant System (AS) was introduced in the early 1990's by Marco Dorigo and his colleagues, and was firstly applied to the traveling salesman problem (TSP). In TSP, a set of cities is given and the distance between each of them is known. The goal is to find the shortest tour so that each city is visited exactly once and the tour ends in the initial city. For a set of cities, we consider d_{ij} to be the distance between any given cities i and j, such that the path length $d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{\frac{1}{2}}$, where (x_i, y_i), (x_j, y_j) is the coordinates of city i and city j. Let τ_{ij} be the amount of pheromone in the edge that connects i and j. Initially, each of m ants is put on some randomly chosen city, and then decides independently which city to go to using a transition rule that is a function of the distance to the city and the amount of pheromone of the present connecting path, until the tour is completed. The probability, which shows the transition rule, of the kth ant making the transition from city i to city j, is given by:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{if } j \in J_i^k \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $\eta_{ij} = 1/d_{ij}$ is a heuristic value, α and β are two parameters which determine the relative importance of pheromone value and heuristic information, and J_i^k is the feasible neighborhood of ant k, that is, the set of cities remains to be visited by ant k positioned on city i. After all ants have built a tour, ants perform following pheromone update rule:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \tag{2}$$

where $\rho \in (0,1)$ is the evaporation rate of the pheromone trail, and $\Delta\tau_{ij}^k$ is the amount of pheromone laid on path (i,j) by the kth ant. The amount of pheromone an ant k deposit on an edge (i,j) is defined by $L^k(t)$, the length of the tour created by that ant at iteration t as follows:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)}, & \text{if } (i,j) \text{ is used by ant } k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where Q is constant [23]. In this way, the increase of pheromone for an edge depends on the number of ants that use this edge, and on the amount of the solutions found by those ants.

Even though the original AS algorithm showed to be a viable method for attacking the TSP problem, it was found to be inferior to state-of-the-art algorithms for the TSP as well as for other problems. Therefore, several extensions and improvements of the original AS algorithm were introduced over the years. In Rank-based AS [15], a number of best ants of the current iteration are allowed to add pheromone and the global-best tour is always used to update the pheromone trails. To this aim the ants are sorted by tour length, and the amount of pheromone an ant may deposit is weighted according to the rank r of the ant. In each iteration, only the $(w-1)$ best-ranked ants and the ant that produced the best-so-far tour are allowed to deposit pheromone. The rth best ant of the current iteration contributes to pheromone updating with a weight given by

$\max\{0, w-r\}$. Additionally, the global best solution, which gives the strongest feedback, is given weight w. Thus the improved update rule is

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{r=1}^{w-1} (w-r) \cdot \Delta\tau_{ij}^{r}(t) + w \cdot \Delta\tau_{ij}^{gb}(t) \tag{4}$$

where $\Delta\tau_{ij}^{r}(t) = Q/L^{r}(t)$ and $\Delta\tau_{ij}^{gb}(t) = Q/L^{gb}(t)$.

3. **Self-Evolving Ant Colony Optimization.** The same as in other heuristic algorithm, the adjustment of the balance between intensification and diversification is one of the most important themes in the study of ACO algorithm. In detail, too much emphasis on the intensification can make ants converge to a local minimum and too much emphasis on the diversification can cause an unstable state. In Section 2, we surveyed the original ACO algorithm and some improvements. We note that only the pheromone information of ants that have good solutions was strengthened and thus these algorithms are apt to converge too early. So most of existing ACO algorithms are aiming to adjust the intensification in depositing pheromone. However, it remains difficult to control the balance between intensification and diversification.

In order to acquire good balance between intensification and diversification, traditional ACO algorithms always try to change the pheromone update rule, such as EAS [14], Rank-based AS [15], MMAS [16-18] and ACS [19,20]. However, in this paper, we adopted the update rule of rank-based AS directly, as the basic pheromone update rule, but not introduce a completely new pheromone update rule. Moreover, a self-evolving mechanism was adopted. In this mechanism, two kinds of pheromone information are used, which are the group pheromone information and the individual information of each ant. The group pheromone information is used for pheromone trail updating by rank-based rule, while the individual information of each ant is employed as a temporary addition of individual information to the updated pheromone trail. As a result, different to rank-based AS, the solution construction of each ant will be affected not only by the overall pheromone amount on each edge after partial evaporation and deposition in a rank-based model, but also by the quality of solution searched by this ant in the last iteration, respectively. So each ant self-evolves by its individual information and the group information.

By tuning $\varepsilon \in (0, 1)$, a parameter which shows the relative influence of group pheromone information and temporary individual pheromone information, the balance of the intensification and diversification is adjusted and evolvement of each ant is expected. In other words, the solution quality of each ant is expected to grow better generation by generation, by the use of the two kind of pheromone information to guide the solution construction.

Through this way, compared with the rank-based AS, the diversification of the method will be greatly affected by the individual ants, so paths of unvisited or relatively unexplored search space regions will get more desirable for the following ants. Thus, instead of quickly converging to a sub-optimal solution, ants will evolve to more promising solution space regions generation by generation.

In order to interpret the proposed method, we introduce the high-level view of the self-evolving ACO in Figure 1 and the outline of the proposed solution construction in Figure 2 where τ_{ij} and τ_{ij}^{k} are used to express the amount of ordinary pheromone trail and the amount of pheromone trail which ant k uses to construct solution.

To exactly describe this algorithm with a mathematical model, we first give the outline of the proposed solution construction in the proposed ACO algorithm in Figure 2. As depicted in this figure, after all of the ants have built solutions in last iteration, the best-so-far solution is updated in the statistic information update step, and pheromone

procedure Self-evolving ACO
  Initialize data
  while (not terminate) **do**
    Construct solutions
    Update statistics
    Update pheromone trails
  end-while
end-procedure

FIGURE 1. Procedure of an ACO algorithm

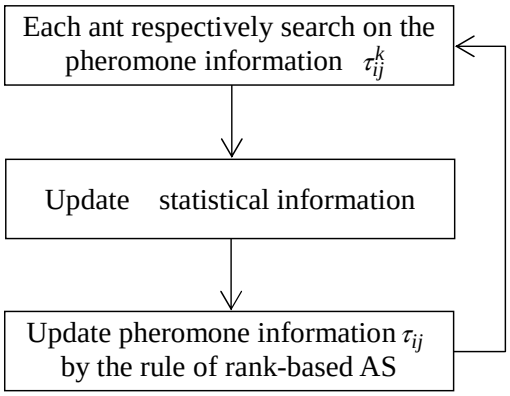

FIGURE 2. Outline of proposed solution construction

information is updated by the rule of Rank-based AS in pheromone information update step while an additional weight $\varepsilon \in (0,1)$ is exploited to adjust the influence of the newly updated rank-based pheromone deposition, which we call it group information τ_{ij}^{G}, upon the solution construction of the next iteration. With this newly adopted parameter $\varepsilon \in (0,1)$, we give the equation of pheromone update as:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \varepsilon \cdot \tau_{ij}^{\mathrm{G}}(t+1) \tag{5}$$

The group information $\tau_{ij}^{\mathrm{G}}(t)$ in Equation (5) can be described as follows:

$$\tau_{ij}^{\mathrm{G}}(t+1) = \sum_{r=1}^{w-1} (w-r) \cdot \Delta\tau_{ij}^{r}(t) + w \cdot \Delta\tau_{ij}^{gb}(t) \tag{6}$$

$$\Delta\tau_{ij}^{r}(t) = \begin{cases} \frac{Q}{L^r(t)}, & \text{if } (i,j) \in T^r(t) \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$\Delta\tau_{ij}^{gb}(t) = \begin{cases} \frac{Q}{L^{gb}(t)}, & \text{if } (i,j) \in T^{gb}(t) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where Q is constant, $L^r(t)$ is the length of the tour generated by the rth best ant in iteration t, $T^r(t)$ is the set of edges constituting the tour, L^{gb} is the length of the tour generated by the global best ant and T^{gb} is the set of edges constituting it.

```
procedure Construct solutions
  Empty memory of each ant
  for each ant do
    Deposit temporary individual pheromone trails
    Assign a random initial city
    while (tour is not completed) do
      Decide the next city on temporary
      individual pheromone information
    end-while
    Evaporate all temporary individual pheromone trails
  end-for
end-procedure
```

FIGURE 3. Outline of proposed solution construction

Once the previous pheromone update step terminates, each ant starts its solution construction on its respective pheromone information τ_{ij}^k, which is only used by tour construction but not used to pheromone update, and can be described as below:

$$\tau_{ij}^k(t+1) = \tau_{ij}(t+1) + (1-\varepsilon) \cdot \tau_{ij}^{I_k}(t+1) \tag{9}$$

$$\tau_{ij}^{I_k}(t+1) = \begin{cases} \frac{Q}{L^k(t)}, & \text{if } (i,j) \in T^k(t) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $\tau_{ij}^{I_k}$, called temporary individual pheromone information in this paper, is the pheromone quantity related with the tour length which is found by the kth ant in last iteration and is temporarily added to current pheromone trails with weight $(1-\varepsilon)$, to obtain respective pheromone on which each ant uses to search solution respectively, and Q is constant, $L^k(t)$ is the length of the tour generated by the kth ant in iteration t. As to detailed procedure on how to execute solution construction in a program, we give Figure 3.

To sum up, based on prior description, we can note that the respective pheromone information, which is used to construct solution by each ant, can be derived by integrating Equation (5), Equation (6) and Equation (9) as follows:

$$\tau_{ij}^k(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \varepsilon \cdot \tau_{ij}^{\mathrm{G}}(t+1) + (1-\varepsilon) \cdot \tau_{ij}^{I_k}(t+1) \tag{11}$$

By Equation (11), we can conclude that the respective pheromone which is used by each ant when building its solution consists of two parts, that is, the left part is the group information and the right part is the temporary individual pheromone information, and that $\varepsilon \in (0,1)$ controls the relative influence of the two kinds of information upon solution construction of each ant.

4. **Simulation Results.** In order to assess the effectiveness of the proposed ACO algorithm, extensive simulations were carried out over some TSPLIB benchmark problems on a PC Station (Intel, 2.66GHz). As the parameter setting of the proposed method, if α is too high compared with β, the ACO algorithm tends to enter stagnation behavior without finding good solutions. If α is too low, the algorithm operates like a repeated construction heuristic and generates good solutions, but can not exploit the positive feedback. The same is true for ρ. If ρ is too close to zero, most of the global information contained in the trail levels evaporates immediately and learning does not take place. If ρ is too close to one, there is the danger of early convergence of the algorithm. The parameter Q measures

the influence of the new information (length of the tours) relative to the influence of the initial trail levels. As long as Q is not too small, this parameter is not crucial for the convergence of the algorithm. The setting $\alpha = 1$, $\beta = 5$, $\rho = 0.5$, $Q = 100$ and $w = 6$ is suggested to be advantageous in [12,15]. In addition, through extensive experiments on the proposed method, we found that if the colony size is too small or too big, the proposed method shows bad performance, so the colony size is set to 30.

4.1. **Parameter value of ε.** In the proposed ACO algorithms, as described in Section 3, parameter ε was adopted to control the relative importance between the group pheromone information and the temporary individual pheromone information. By tuning this parameter, the relative degree of exploration and exploitation was adjusted to achieve best performance, that is to say, the performance of the proposed method is affected by the choice of whether to concentrate the search of the system around the best-so-far solution or to explore other tours in the search process. Too small value of ε leads to less emphasis of group experience and temporary individual experience dominates in the search process and as a result, it is difficult to find the optimal results because of rapid change of the search space. On the other hand, too large value of ε causes that at the initial stage, ants are apt to concentrate around some local optimal solution, resulting in a stagnation situation. To see the influence of this parameter upon the performance, we tested the proposed ACO algorithm on problem Oliver30 including 30 cities, problem eil50 including 50 cities, problem eil75 including 75 cities, and problem kroA100 including 100 cities using different values of parameter ε. For each different value of ε of each problem, 100 runs were performed. The simulation results on different TSP instances are summarized in Table 1. In this table, we give the rates to find the optimal solutions when

Table 1. Simulation results

ε	oliver30	eil50	eil75	kroA100
0	2%	0%	0%	0%
0.025	50%	2%	4%	2%
0.075	100%	16%	54%	23%
0.125	100%	77%	90%	100%
0.175	99%	60%	48%	38%
0.225	49%	39%	30%	27%
0.275	28%	9%	16%	7%
0.325	25%	7%	13%	3%
0.375	24%	4%	12%	1%
0.425	17%	3%	4%	0%
0.475	16%	2%	5%	0%
0.525	16%	1%	4%	0%
0.575	18%	0%	3%	0%
0.625	12%	1%	1%	0%
0.675	9%	2%	0%	0%
0.725	8%	1%	0%	0%
0.775	7%	0%	0%	0%
0.825	5%	0%	0%	0%
0.875	2%	0%	0%	0%
0.925	2%	0%	0%	0%
0.975	1%	1%	0%	0%
1	1%	1%	0%	0%

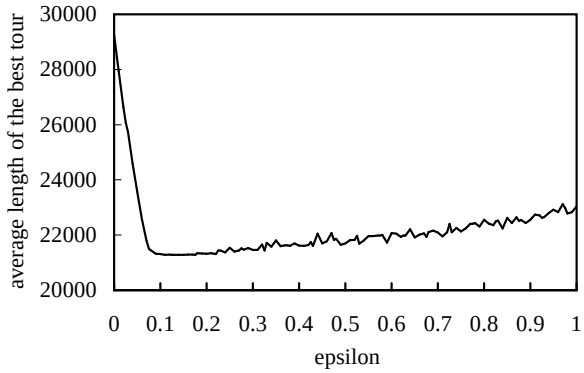

FIGURE 4. Average tour length using different ε

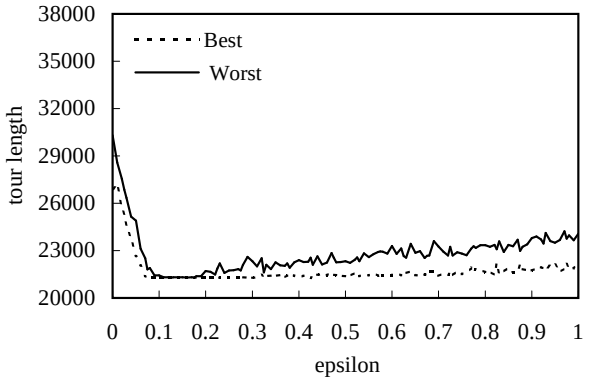

FIGURE 5. Best and worst tour length using different ε

using different ε on these TSP instances. From this table we note that when $\varepsilon = 0$, that is, there is no group pheromone information to affect the solution construction, the rates to find the optimal solutions are very low or even 0, and when $\varepsilon = 1$, that is, there is no individual pheromone information to affect solution construction, the rates are also very low or even 0.

Through observing this table, we can conclude a characteristic of our proposed algorithm, that is, the proposed algorithm can find the optimal solution with the highest rate when ε is around 0.125. In addition to the rate to find the optimal solution, we can also assess the effectiveness of the algorithm under a specific ε by the average length of the best tour, or the difference between the best and the worst tour. As we know that the smaller the average length of the best tour or the difference between the best and the worst tour is, the better the the algorithm performs, we present Figure 4 and Figure 5, in which the results of the simulation carried out over problem kroA100 are given. In Figure 4, we give average length of the best tour when using different ε, and in Figure 5 we give the best and worst tour lengths when using different ε. From both figures, we find that the proposed algorithm can get the smallest average length of the best tour and even the same value of the best and worst tour when ε is around 0.125. As described above, the proposed algorithm shows the best performance when ε locates around 0.125. In following simulations, we set the value of ε with 0.125.

4.2. **Search ability of the proposed algorithm.** As described in Section 3, different to the traditional ACO algorithms, two kinds of different pheromone information were

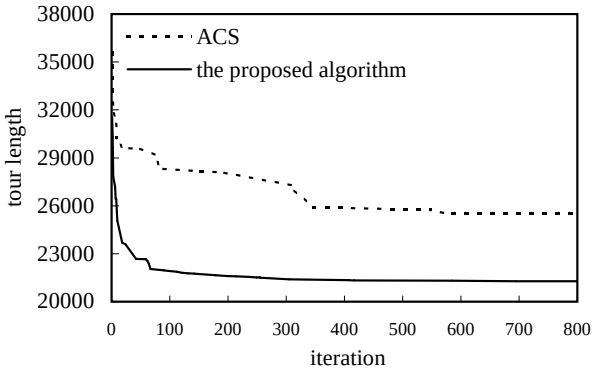

FIGURE 6. Tour length varying with iteration

introduced in this paper. Because the group pheromone information controls the intensification and the individual pheromone information affects the diversification of the algorithm, the balance of intensification and diversification can be easily adjusted by tuning the relative influence of the two kinds of pheromone information, while the traditional ACO algorithms have no quantitative parameter like ε in the proposed method to adjust the balance of intensification and diversification.

From experiment result of Section 4.1, we obtain the optimal value of ε. To see the search ability of the proposed algorithm, we compared the evolving process of the tour length of the proposed algorithm with a conventional ACO algorithm called ACS [19,20], an extension of the first ACO, through developing simulations on problem kroA100 respectively as well. The experimental simulation results are given by Figure 6. The figure shows that the proposed algorithm converged to a value very close to the optimal tour length after 400 iterations and to the optimal length after 695 iterations, while the conventional method converged very slowly and to a very large value that is far away from the optimal value even after 800 iterations. In addition, the curve of the proposed method is smoother than the traditional method. So we can conclude that the self-evolving mechanism takes effect indeed. In other words, the better search ability of the proposed algorithm is mainly due to the self-evolvement of each ant by exploiting individual information that gradually affects the solution construction iteration by iteration based on the rank-based pheromone deposition.

4.3. **Comparison with other algorithms.** In addition to the problem kroA100, we also tested the proposed ACO algorithm on some other TSPLIB benchmark problems to see the ability of the global search and local converge of the proposed algorithm. For each of instances, 100 simulation runs were performed. The results are shown in Table 2 where the results produced by the ACS [19,20], genetic algorithm (GA), evolutionary programming (EP), simulated annealing (SA) are also listed for comparison. We report the best integer tour length, the best real tour length (in parentheses) and the number of tours required to find the best integer tour length (in square brackets). The difference between integer and real tour length is that in the first case distances between cities are measured by integer numbers, while in the second case by floating point approximations of real numbers. Table 2 shows that for the integer results the proposed algorithm can acquire the optimal results using the least calculation time, and for the results of real lengths, the proposed algorithm can also find the smallest solutions. Through observing the table, it is easy to note that the proposed ACO algorithm outperforms other algorithms in both the solution quality and the computation cost.

TABLE 2. Simulation results

Problem	Optimum	GA	EP	SA	ACS	Proposed algorithm
Oliver30 (30-city)	420	421	420	424	420	420
	423.74	(N/A)	423.74	(N/A)	423.74	423.74
		[3 200]	[40 000]	[24 617]	[830]	[199]
Eil50 (50-city)	425	428	426	443	425	425
	(N/A)	(N/A)	427.86	(N/A)	427.96	427.86
		[25 000]	[100 000]	[68 512]	[1 830]	[360]
Eil75 (75-city)	535	545	542	580	535	535
	(N/A)	(N/A)	549.18	(N/A)	542.31	542.31
		80 000	[325 000]	[173 250]	[3 480]	[597]
KroA100 (100-city)	21 282	21 761	N/A	N/A	21 282	21 282
	(N/A)	(N/A)	(N/A)	(N/A)	21 285.44	21 285.44
		[103 000]	[N/A]	[N/A]	[4 820]	[695]

5. **Conclusions.** An improved ACO algorithm called self-evolving ant colony optimization for efficiently solving combinatorial optimization problems was proposed in this paper. In the proposed ACO algorithm, the balance between intensification and diversification can be adjusted by the gradually self-evolving of each ant, using the temporary individual information of last iteration and the group information that applied rank-based pheromone-update rule. The proposed ACO algorithm was evaluated experimentally through simulating some TSP benchmark problems. From the simulation results, we noted that the proposed improved ACO has superior performance compared with the original Rank-based AS and some other methods. It is worth noting that the idea to adjust the balance between intensification and diversification can also be applied to other ACO algorithms.

REFERENCES

[1] C. Blum and A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Computing Surveys*, vol.35, no.3, pp.268-308, 2003.

[2] S. D. Shtovba, Ant algorithms: Theory and applications, *Programming and Computer Software*, vol.31, no.4, pp.167-178, 2005.

[3] C. Blum, Ant colony optimization: Introduction and recent trends, *Physics of Life Reviews*, vol.2, no.4, pp.353-373, 2005.

[4] S. S. Kim, I.-H. Kim, V. Mani and H. J. Kim, Ant colony optimization for SONET ring loading problem, *International Journal of Innovative Computing, Information and Control*, vol.4, no.7, pp.1617-1626, 2008.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*, The MIT Press, 2004.

[6] J. Bell and P. Mcmullen, Ant colony optimization techniques for the vehicle routing problem, *Advanced Engineering Informatics*, vol.18, no.1, pp.41-48, 2004.

[7] S. S. Kim, I.-H. Kim, V. Mani, H. J. Kim and D. P. Agrawal, Partitioning of mobile network into location areas using ant colony optimization, *ICIC Express Letters, Part B: Applications*, vol.1, no.1, pp.39-44, 2010.

[8] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini and M. Trubian, Heuristics from nature for hard combinatorial optimization problems, *International Transactions on Operational Research*, vol.3, no.1, pp.1-21, 1996.

[9] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck and B. Baesens, Classification with ant colony optimization, *IEEE Transactions on Evolutionary Computation*, vol.11, no.5, pp.651-665, 2007.

[10] C. Twomey, T. Stützle, M. Dorigo, M. Manfrin and M. Birattari, An analysis of communication policies for homogeneous multi-colony ACO algorithms, *Information Sciences*, vol.180, no.12, pp.2390-2404, 2010.

[11] M. Dorigo, M. Birattari and T. Stützle, Ant colony optimization, *IEEE Computational Intelligence Magazine*, vol.1, no.4, pp.28-39, 2006.

[12] M. Dorigo, V. Maniezzo and A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics*, vol.26, no.1, pp.29-41, 1996.

[13] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[14] M. Dorigo and C. Blum, Ant colony optimization theory: A survey, *Theoretical Computer Science*, vol.344, no.2-3, pp.243-278, 2005.

[15] B. Bullnheimer, R. Hartl and C. Strauss, A new rank based version of the ant system, *Central European Journal of Operations Research*, vol.7, no.1, pp.25-38, 1999.

[16] T. Stützle and H. H. Hoos, MAXMIN ant system, *Future Generation Computer Systems*, vol.16, no.9, pp.889-914, 2000.

[17] T. Stützle and M. Dorigo, A short convergence proof for a class of ACO algorithms, *IEEE Transactions on Evolutionary Computation*, vol.6, no.4, pp.358-365, 2002.

[18] R. Pitakaso, C. Almeder, K. F. Doerner and R. F. Hartl, A MAX-MIN ant system for unconstrained multi-level lot-sizing problems, *Computers and Operations Research*, vol.34, no.9, pp.2533-2552, 2007.

[19] M. Dorigo and L. M. Gambardella, Ant colonies for the traveling salesman problem, *BioSystems*, vol.43, no.2, pp.73-81, 1997.

[20] M. Dorigo and L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, vol.1, no.1, pp.53-66, 1997.

[21] G. Reinelt, A traveling salesman problem library, *ORSA Journal on Computer*, vol.3, no.4, pp.376-384, 1991.

[22] L. M. Gambardella and M. Dorigo, Solving symmetric and asymmetric TSPs by ant colonies, *Proc. of the IEEE Conference on Evolutionary Computation*, New York, pp.622-627, 1996.

[23] S. D. Shtovba, Ant algorithms: Theory and applications, *Programming and Computer Software*, vol.31, no.4, pp.167-178, 2005.